National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

UNIVERSITY OF ALBERTA

Adaptive Sampling with Predictive Encoding – ASPEN

By

Vikas Nehru ©

A thesis
submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree
of Master of Science

Department of Computing Science

Edmonton, Alberta
Fall 1994

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-95086-2

Canada

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Vikas Nehru

TITLE OF THESIS: Adaptive Sampling with Predictive Encoding – ASPEN

DEGREE: Master of Science

YEAR THIS DEGREE GRANTED: 1994

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

(Signed) _____

Permanent Address:
Flat No. 8734, Vasant Kunj,
Pocket 8, Sector C
New Delhi, India.

Date: August 5th, 1994

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled *Adaptive Sampling with Predictive Encoding – ASPEN* submitted by *Vikas Nehru* in partial fulfillment of the requirements for the degree of Master of Science.

Supervisor: Dr. X. Li

Examiner: Dr. J. Harms (Computing Science)

External: Dr. W. Joerg (Electrical Engineering)

Date: August 4th, 1994.

This thesis is dedicated to the ones I love.

Beloved Wife – Ritu,

Brother – Vishal,

Parents – Jawahar & Krishna.

Abstract

In the recent years, images have played an increasingly important role in our society. Today, they are being used in a variety of applications, such as advertising, medical diagnosis, tracking, weather forecasting and television. However, the large information content of images coupled with the limited resources available for storage and transmission, create the need for image compression.

This thesis presents a *general purpose* image compression scheme – *Adaptive Sampling with Predictive Encoding* (ASPEN), which is essentially based on the Predictive Compression model and Detail-based segmentation of images. ASPEN's functionality provides for either lossless or lossy compression, and it can also be tailored to meet the particular needs of different applications.

A picture usually comprises of areas of varying detail. ASPEN first attempts to detect these areas, and then it samples key points from them. An area with greater detail requires many sample points to suitably represent it, whereas regions with lesser detail can be represented with fewer points. These key sample points are then used to predict the remainder of the image. Predictive encoding schemes are able to directly exploit the dependency between neighbouring pixels within an image and hence are effective in reducing the interpixel redundancy. Combining the use of adaptive sampling with predictive encoding, ASPEN manages to achieve compression while maintaining the desired image quality. The performance of a prototype version of ASPEN is compared to other compression techniques in terms of compression ratio and image quality. Some applications for which ASPEN is particularly suited are also outlined.

Acknowledgements

I would like to take this opportunity to sincerely express my appreciation to those who have helped me during the duration of my graduate studies.

I am especially thankful to Dr. Xiaobo Li, my supervisor, for all his guidance, encouragement and patience throughout this research. I could not have asked for a better *Guru*. I am also grateful to members of my committee, Dr. Janelle Harms and Dr. Warren Joerg for their valuable comments which made this work complete.

Many many warm thanks to all my wonderful friends. Rohit, Vinay, Anoma, Salima, Anil, Anila, Sathiya, Gulam, Shafik, Shwetha, Simmi, Rupa and Ravi. I could not have survived without your friendship. Very special thanks to all my friends at school; Aditya, Ramesh, Pavan, Shurjo, Shankar, Kaladhar, Naren, Srinivas, Kavita, Chiradeep, Srini, Kevin, Dennis, Michael, Magued, Iqbal, Gregor, Sydney, Kit, and Pius. You guys made life in the department a whole lot livelier.

Without the love and support from my parents, I would have never made it this far. Special thanks to my mother for all her prayers and affection, and also to my father and brother for their love and encouragement.

Finally, and most of all, I would like to thank my cherished wife *Ritu*, for her understanding, sacrifice and encouragement throughout the completion of this dream.

Contents

1 Introduction 1

1.1 Compression Algorithms & Standards 2

1.2 Adaptive Sampling with Predictive Encoding – (ASPEN) 3

1.3 Organization of the thesis 4

2 Image Compression Methods 5

2.1 Entropy and Redundancy 6

2.2 Encoding Techniques 7

2.2.1 Huffman Coding 7

2.2.2 Arithmetic Coding 8

2.2.3 Dictionary Coding 8

2.3 Still-Image Compression Schemes 9

2.3.1 Traditional Linear Predictive Coding – LPC 9

2.3.2 Joint Photographic Experts Group – JPEG 11

2.3.3 Vector Quantization 12

2.3.4 Fractal Compression 13

2.3.5 Pyramidal Compression 13

2.3.6 Experiments with Traditional LPC 14

2.4 Comparison of Compression Methods 14

3 Description of ASPEN 15

3.1 Iterative Segmentation 16

3.1.1 Demonstration of Sampling 19

3.1.2 Data Structure to Represent Segmentation 20
3.2 Prediction Schemes 20
3.2.1 Bi-Linear Interpolation 21
3.2.2 Two-Phase Interpolation 22
3.2.3 Two-Phase-Extended Interpolation – (2PE) 24
3.2.4 Splinal Interpolation 25
3.2.5 Correlation-based Prediction 26
3.2.6 Refined Coefficients 28
3.3 Quantization 29
3.3.1 Uniform Quantization 29
3.3.2 Optimal Quantization 30
3.3.3 Optimal versus Uniform Quantization 31
3.3.4 Adaptive Quantization 32

4 Search in Parameter Space **34**
4.1 First Phase – Wide Range Search 35
4.2 Second Phase – Fine Tuning 39

5 Performance Comparisons **42**
5.1 Performance Measures 42
5.1.1 Compression Ratio 42
5.1.2 Fidelity Criteria 43
5.1.3 Compression Time 44
5.2 Characteristics of the Tests 45
5.3 ASPEN vs LPC 47
5.3.1 Analysis of the Results: ASPEN vs LPC 48
5.4 ASPEN vs JPEG 48
5.4.1 Analysis of the Results: ASPEN vs JPEG 49
5.5 Examples 50

6 Possible Applications of ASPEN **51**
6.1 Scanning an Image Database 51

6.2 Combining Image Encryption with Compression 52
6.3 Redundancy Sieving 53
6.4 Like-Image Database 54

7 Conclusions **55**
7.1 Suggestions for Future Work 56

Bibliography **58**

List of Tables

3.1 **Optimal versus Uniform Quantization** 31

4.1 **Setting of ASPEN parameters in the First Phase** 36
4.2 **Setting of ASPEN parameters in the Second Phase** 39
4.3 **Optimal Setting of ASPEN parameters** 41

5.1 **Different Images used for Testing** 46
5.2 **Setting of ASPEN parameters for comparisons** 47
5.3 **Comparison of ASPEN and LPC** 47
5.4 **Comparison of ASPEN and JPEG** 49

List of Figures

2.1 General Compression Algorithm 5
2.2 Differential Pulse Code Modulation 9
2.3 Matrix f – A digitized picture 10
2.4 Baseline Sequential JPEG Algorithm 11

3.1 Overview of Adaptive Sampling with Predictive Encoding 15
3.2 Demonstration of sampling 19
3.3 Data Structure to represent Segmentation. 20
3.4 Prediction of a Block. 21
3.5 Bi-Linear Interpolation. 21
3.6 Two-Phase Interpolation. 23
3.7 Two-Phase-Extended Interpolation. 25
3.8 Difference between Linear and splinal Interpolation. 26
3.9 Coefficient based Prediction. 26
3.10 Refined Coefficient. 28
3.11 Uniform Quantization. 29
3.12 Optimal Quantization. 30
3.13 Examples of Optimal and Uniform Quantization 32
3.14 Adaptive Quantization. 33

4.1 Table represented as an image 36
4.2 Demonstration of Trend – Image1 – Wide Search 37
4.3 Demonstration of Trend – Image2 – Wide Search 38
4.4 Demonstration of Trend – Image3 – Wide Search 38

4.5 **Table represented as an image** 39
4.6 **Demonstration of Trend – Fine Tuning** 40

5.1 **Examples of Images** 46
5.2 **Examples of Compression – ASPEN and JPEG** 50

6.1 **Blockiness in JPEG at low compression levels** 52
6.2 **Key for Image Reconstruction** 53
6.3 **Redundancy Sieving in ASPEN** 54

Chapter 1

Introduction

Interest in image processing techniques dates back more than half a century, when the first digitized images were transmitted across the Atlantic [3]. The limited speed and storage capabilities of those early computers, however, were a roadblock to the widespread growth of imaging applications. The recent improvements in high-speed network technology coupled with the increased speed and storage capabilities of computers have provided the necessary impetus for rapid growth in this field. We are now able to manipulate images in ways heretofore only imagined. Today, it is not uncommon to have workstations capable of displaying full-motion colour video in real-time.

However, the large information content of digital images, requires immense computing resources for processing, storage and transmission. Let us consider a single picture of dimensions 512 x 512, that consists of 262,144 pixels. If the image is 24-bit colour, it requires 786,432 bytes or just under 1 Mega byte of storage. Transmission of this single digital image over a link with 9600 bps would take about 1.7 minutes (including overhead). Real-time transmission of Digital Video requires 30 frames per second, which equates to about 30 Mbytes per second. In High Definition Television (HDTV) where image resolution is even higher, this figure is about 150 Mbytes per second. The present fibre-optic network technology due to bottlenecks such as network access and processing time, is not able to meet this requirement [5, 2, 18]. Clearly some method of compression is necessary to reduce the volume of bits required to represent

the information in an image.

Image compression techniques are quite diverse, ranging from predictive to transform based methods. All compression schemes take a raw digital image and convert it into a format which requires fewer bits to transmit or store. Compression is a 2-stage process. At one end, an encoder compresses the image into (hopefully) a smaller file. The file is then stored (or transmitted) and reconstructed at the other end by a decoder. A system that performs both these operations is referred to as a *codec* (coder-decoder).

1.1 Compression Algorithms & Standards

Algorithms

Compression algorithms can be classified into two categories: lossless and lossy, depending on the method's effect on the data being compressed. A lossless scheme, such as traditional linear predictive coding, is able to reconstruct the original image *exactly* after compression. Lossless compression, however, is not very efficient. A 4:1 compression ratio is considered very good [16].

On the other hand, a lossy method such as JPEG, alters the data during compression. Generally, lossy schemes can attain high compression ratios without a perceptible change in the uncompressed image quality (visual quality of the image is a subjective criterion). Lossy compression schemes are suitable for most non-critical image processing applications, such as teleconferencing or mug-shot databases, where some loss in image quality is acceptable. Lossless compression schemes must be used in applications where even slight visible errors are undesirable, such as medical images for diagnosis.

Standards

Research community and the industry, have defined many standards for image compression, each of which deals with different types of images and applications. The JPEG (Joint Photographic Experts Group) standard is designed for the compression of still images (grayscale or colour). For the lossless compression of binary (one-bit/pixel) images, a standard called JBIG (Joint Bi-Level Image Experts Group) has been devel-

oped. MPEG (Motion Picture Expert Group) generates standards for digital video (sequence of frames in time) and audio compression. Some details of several image compression schemes and standards will be presented in the next chapter.

1.2 Adaptive Sampling with Predictive Encoding – (ASPEN)

Detail-based segmentation of images and the Predictive compression model are the essential underlying principles of ASPEN. In segmentation-based image compression, the image is segmented, that is the image pixels are divided into mutually exclusive spatial regions based on some criteria [9, 14].

ASPEN first attempts to segment areas of the image based on their *detail-content*. A high detail-content implies that the region has rapidly varying gray levels, whereas regions with low detail-content have almost constant or slowly varying gray levels. ASPEN segments a region with considerable detail (samples more points from it) till the region can be suitably predicted from the sample points or when it cannot be segmented any further. Thus, areas with greater detail have more points sampled from them and regions with lesser detail are represented with fewer points. These key sample points are then used to predict the remainder of the image using a variety of prediction methods.

ASPEN has a number of distinct advantages:

- ASPEN adapts itself to each image – automatically detecting areas of detail and then only sampling *key* points (that are most representative).
- Adaptive sampling followed by predictive encoding can achieve the desired compression ratio and image quality.
- Being a predictive encoding scheme, ASPEN can be utilized for either lossless or lossy compression.

ASPEN can be used to compress different images within several applications. For images requiring lossless compression such as medical images to teleconferencing im-

ages for lossy compression, ASPEN is capable of adapting to each image and particular needs of the application.

1.3 Organization of the thesis

Chapter 2 presents fundamental knowledge about image compression, followed by a description of Traditional Linear Predictive Coding and JPEG compression methods. Chapter 3 describes the design and implementation of a prototype version of ASPEN. Experiments designed to search the ASPEN parameter space for a *good* setting, are detailed in chapter 4. Performance results from the comparison of ASPEN with JPEG and Traditional Linear Predictive Coding are presented in chapter 5. Some applications for which ASPEN is particularly suited are outlined in chapter 6. Conclusions are presented in chapter 7 along with some outlines for future research directions.

Chapter 2

Image Compression Methods

The large amount of data required to represent an image heavily taxes the computing resources (cpu and memory), in turn limiting the power of image processing. Advances in technology permit ever increasing image resolution (spatially and in gray-levels), thereby creating the need to limit the resulting data volume. By removing the redundancy present in most images, compression algorithms reduce the number of bits needed to represent an image in a way which makes reconstruction possible.

A general algorithm for data compression is shown in figure 2.1.

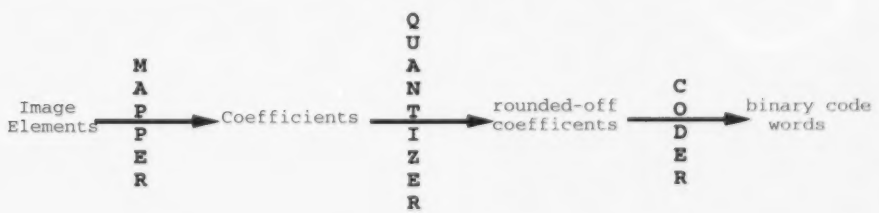


Figure 2.1: **General Compression Algorithm**

The image data are first mapped from the pixel domain to another domain (frequency or spatial), using either Transform, Predictive or Hybrid compression schemes. This step aims to reduce the information redundancy caused by high correlation (dependency amongst pixels) present in the image data. The quantizer rounds off each mapped datum to one of a smaller number of possible values. Most reduction in bit volume is achieved in this phase, at the expense of information loss. Finally the

quantizer output is coded using either a variable or a fixed length code. Variable length codes use shorter codes for more frequent data, thereby increasing compression efficiency, whereas fixed length codes offer ease of handling.

2.1 Entropy and Redundancy

In order to pick an appropriate image compression technique, image data properties such as entropy and redundancy must be determined first.

Entropy is a measure of the degree of randomness in a set of random variables [3]. In coding applications, entropy represents the amount of information associated with the set of coder input values. It gives a lower bound on the average number of bits required to code those inputs. If an image has G gray levels and the probability of occurrence of gray level k is $P(k)$, then entropy H is defined as

$$H = -\sum_{k=0}^{G-1} P(k) \cdot \log_2(P(k)). \tag{2.1}$$

For any image, it is not possible to code the set of gray levels using less than H bits on average. Therefore, the concept of entropy provides a performance criterion against which we can measure any particular code.

The limit of *maximum achievable compression ratio - K* then is,

$$K = \frac{b}{H}. \tag{2.2}$$

where b is the least number of bits needed to represent the image quantization levels. Theoretical limits of possible image compression can be found using formula 2.2. For example, let the entropy of an image be $H = 3$, where the image pixels are quantized into 256 gray levels or 8 bits per pixel. This entropy value implies that the image data can be represented by at most 3 bits per pixel without any loss of information. In this case, the maximum compression any code would achieve is $K = 8/3 = 2.6$.

Information **redundancy** (r) is a central issue in digital image compression and is defined as,

$$r = b - H. \tag{2.3}$$

Three forms of redundancies can be exploited in still-image compression, and ASPEN particularly deals with only interpixel redundancy.

- **Coding Redundancy**

  If the gray levels of an image are coded in a way that uses more code symbols than absolutely necessary to represent each gray level, the resulting image is said to contain *Coding Redundancy*. Assigning variable length codes to most and least probable gray values removes coding redundancy.

- **Interpixel Redundancy**

  Since the value of any pixel in an image can be reasonably predicted from the value of its neighbours, the information carried by individual pixels is relatively small. Redundancy existing in images because of the correlation present within pixels is termed *Interpixel Redundancy*.

- **Psycho-Visual Redundancy**

  Human eye does not respond with equal sensitivity to all visual information. Certain information simply has less relative importance than other information in normal visual processing. This information is said to be *psychovisually redundant*.

2.2 Encoding Techniques

All encoding techniques use special codes to represent bytes or groups of bytes, which in turn substantially decreases the bit volume required to code the information. Effective encoding schemes aim to exploit patterns or special characteristics (skewed distribution of values) in the data set. Some key encoding schemes are discussed briefly.

2.2.1 Huffman Coding

Huffman coding is a statistical lossless compression scheme which removes coding redundancy in the data stream [6]. It saves on the average number of bits required to code a message by assigning shorter code words to more frequently appearing items.

Therefore, it can afford to assign longer bit strings to less frequently appearing items. A Huffman code can be built in the following manner:

- Sort all items in order of probability of occurrence.
- Create a tree by successively combining two items of the lowest probability to form a new composite item; probability of the new item is the sum of its parts.
- Beginning at the topmost node of the tree, label each leaf with a "1" if its probability is higher than that of the other leaf attached to the same node, and a "0" otherwise.

This algorithm produces a unique code string for each data item. Since no code string is the prefix substring of any other code, no separators are needed between strings in order to perform decoding. This technique is used in most archivers (pkzip, zoo ...).

2.2.2 Arithmetic Coding

Arithmetic coding is also a statistical compression scheme, that works by representing a number by an interval of real numbers between 0 and 1 [13]. Each unique data item is assigned a subinterval of [0,1], proportional to its probability of occurrence. To encode a string of items, subintervals of the interval assigned to the first item in the string are assigned to each data item. Next subintervals are assigned again to the interval corresponding to the second item in the string. This is repeated until the last item in the string is reached. Any real number in this final interval can be used to represent the original string.

2.2.3 Dictionary Coding

The Dictionary or Substitutional class of compressors owe much of their existence to the work of Jakob Ziv and Abraham Lempel in the late 70's (LZ78). The basic idea behind Dictionary compression is to replace an occurrence of a particular phrase or a group of bytes in a set of data, with reference to a previous occurrence of that phrase.

LZ78-based schemes work by entering phrases into a dictionary and when a repeat occurrence of that particular phrase is found, the corresponding dictionary index is

used instead of the phrase. Several compression algorithms are based on this principle, differing in the manner in which they manage the dictionary. Terry Welch's LZW scheme [22], used in the UNIX "compress" program and the Graphic Interchange Format (GIF) is the most well known.

The success of an encoding scheme depends on how well it matches the structure of a given image. Therefore, the structure of the image must be determined first, followed by choosing an encoding method that best fits that structure. However, it is difficult to understand the structural properties inherent in pictorial data. Hence, the design of an image encoder involves a certain amount of experimentation.

2.3 Still-Image Compression Schemes

Any of the above mentioned encoding schemes can be used for image compression. However, since pixels within images usually do not follow any given pattern, encoding schemes on their own do not have much success in compressing images. Therefore, other techniques discussed below have to be used in addition to encoding schemes.

2.3.1 Traditional Linear Predictive Coding – LPC

The Traditional Linear Predictive Coding scheme is based on the Differential Pulse Code Modulation (DPCM) model, in which a pixel is predicted from a combination of pixels already coded in the image.

Original Set of Pixel Values

128	129	130	127	125	128	127	129	130
128	−1	−1	3	2	−3	1	−2	−1

Difference Set

Figure 2.2: **Differential Pulse Code Modulation**

For example, assuming that the coding is done from left to right, the pixel to the left can be used as an estimate for the current pixel, as shown in Fig 2.2. Then the descriptor

fed to the encoder need only be the difference between the pixel being coded and the pixel preceding it. Since in a continous-tone image the differences between one pixel and the next are likely to be small, it is more efficient to encode the difference values than to encode each pixel independently.

In the Traditional scheme, points immediately above and to the left are used as predictors for estimating each pixel value. Fig 2.3 shows a digitized picture represented by matrix f. Let $f(m,n)$ be the element of this matrix that is in the m_{th} column and the n_{th} row.

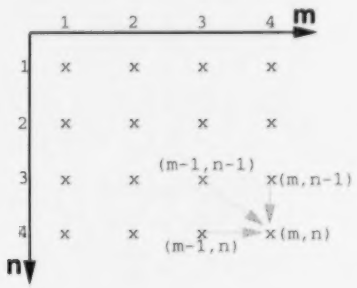


Figure 2.3: **Matrix f – A digitized picture**

The pixel at (m,n) is predicted using points $(m-1,n-1)$, $(m,n-1)$ and $(m-1,n)$. $\hat{f}(m,n)$ – *the predicted value* and $e(m,n)$ – *the error value* are:

$$\hat{f}(m,n) = a_1 f(m-1,n) + a_2 f(m-1,n-1) + a_3 f(m,n-1). \quad (2.4)$$

$$e(m,n) = f(m,n) - \hat{f}(m,n). \quad (2.5)$$

where a_1, a_2, a_3 are image prediction model parameters and are set to minimize the total estimation error, either $\sum |e(m,n)|$ or $\sum e^2(m,n)$. These parameters are calculated based on the correlation of pixels in the image.

It is apparent that if the gray levels in an image are known in the topmost row and the leftmost column, the entire image may be reconstructed from the differential data $e(m,n)$. This data for an image represents a *smaller amount of information* than the gray levels in the original digitized image. This is because images generally contain areas of slowly varying gray levels, and in these regions $e(m,n)$ is very small.

Predictive compression algorithms are described in detail in [16, 11]. Many modifications of predictive compression methods can be found in the literature, some of them also combine predictive compression with other coding schemes [1, 25].

2.3.2 Joint Photographic Experts Group – JPEG

The *Joint Photographic Experts Group* (JPEG – pronounced jay peg) has been working under International Organization for Standardization (ISO), the International Telegraph and Telephone Consultative Committee (CCITT) and the International Electrotechnical Commission (IEC) – for the purpose of developing a standard for colour image compression [12]. This standard describes a family of image compression techniques amongst which *the Baseline Sequential Lossy* and *the Lossless scheme* appear to be the most important.

The JPEG lossless standard is based on the Traditional Linear Predictive model, in which the value of a pixel is estimated from its surrounding neighbours. This estimation (predicted value) is then subtracted from the actual value and the difference is encoded using either Huffman or Arithmetic coding methods. Lossless codecs typically produce around 2:1 compression for colour images with moderately complex scenes [21].

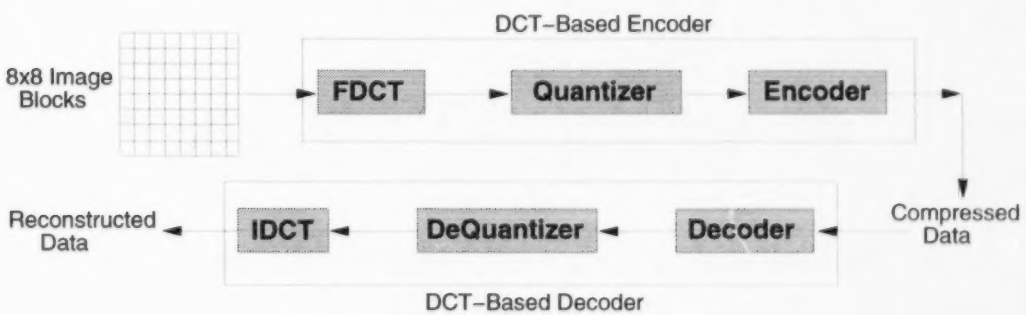


Figure 2.4: **Baseline Sequential JPEG Algorithm**

The JPEG Baseline Sequential Lossy scheme is based on the Discrete Cosine Transform (DCT). The basic algorithm is described below.

First, the range of colours is shifted from $[0, 2^n]$ to $[-2^{n-1}, 2^{n-1}-1]$. Next, the image

is subdivided into 8 x 8 blocks and the Forward DCT is applied to each of them. The DCT is related to Fourier transforms, and transforms images from the spatial domain to the frequency domain.

The FDCT used in JPEG is:

$$F(u,v) = \frac{1}{4}C(u)\cdot C(v)\sum_{x=0}^{7}\sum_{y=0}^{7} I(x,y)\cdot \cos\frac{(2x+1)u\pi}{16}\cdot \cos\frac{(2y+1)v\pi}{16} \quad (2.6)$$

where $C(u), C(v) = \frac{1}{\sqrt{2}}$ for $u, v = 0; C(u), C(v) = 1$ otherwise, I is the input image.

The resulting transformed block is then quantized and coded using a Huffman or Arithmetic encoder. To reconstruct a compressed image, the coded data is first decoded, the Inverse Discrete Cosine Transform (IDCT) is applied to the 8 x 8 blocks and the resulting values are shifted back to the original colour range.

The IDCT used in JPEG is:

$$f(x,y) = \frac{1}{4}C(u)\cdot C(v)\sum_{u=0}^{7}\sum_{v=0}^{7} F(u,v)\cdot \cos\frac{(2x+1)u\pi}{16}\cdot \cos\frac{(2y+1)v\pi}{16} \quad (2.7)$$

JPEG has been designed to exploit known limitations of the human eye, notably the fact that small colour details are not perceived as well as small details of light-and-dark areas. Thus JPEG is intended for compressing images that will be looked at by humans (not suitable for machine interpretation). Furthermore, JPEG works well on images containing natural or real-world scenes where colours change gradually. Because of the nature of DCT-quantization, JPEG is not particularly adept at handling cartoon type images (containing narrow colour transitions within large areas of the same colour). The sharp lines contained in these images can become badly blurred.

A useful property of JPEG is that the degree of lossiness can be varied by adjusting the compression parameters. The image maker can trade off file size against output image quality. However, at high compression rates, images start appearing *blocky* (because of applying quantization to 8 x 8 blocks).

2.3.3 Vector Quantization

The basic idea behind Vector Quantization is Shannon's Rate-Distortion Theory, which states that a better compression performance can always be achieved by coding vectors

instead of scalars.

Applying this idea, Vector Quantization first divides the image into small blocks each of which is represented as a vector [4]. These data vectors are then coded using unique codewords from a codeword dictionary. The premise of the dictionary idea is that some patterns occur more frequently than others in an image. Therefore, it is more efficient to store these patterns in a Codeword dictionary. The vector codes are stored or transmitted along with the codeword dictionary. The advantage of Vector Quantization is a simple receiver structure consisting of a look-up-table, but at the expense of increased complexity at the coder.

2.3.4 Fractal Compression

Traditional geometry with its straight lines and smooth surfaces is not adequate to represent natural objects such as trees, clouds or mountains. Fractal geometry with its convoluted coastlines is able to model such objects more efficiently [8]. Fractal Compression, based on fractal geometry, attempts to represent images by developing models based on fractal equations.

Fractal Image compression is a promising new technology, arguably superior to JPEG. However, it takes a considerable amount of time to compress images. On the other hand, JPEG can be used for real time compression of images. To become widely accepted fractal compression will have to improve its speed of compression.

2.3.5 Pyramidal Compression

Substantial reduction in bit volume can be achieved by representing an image as a pyramid [15]. In pyramidal based schemes, large image areas of the same gray level are represented in higher levels of the pyramid (quad-tree), without the necessity of including lower level nodes in the image representation.

Pyramidal compression can be applied to the idea of Progressive image transmission, which is based on the fact that transmitting the entire image data may not be necessary under some circumstances. Therefore, in progressive transmission, the higher pyramid levels (lower resolution) are transmitted first. The details, that is the

lower pyramid levels follow later.

2.3.6 Experiments with Traditional LPC

Traditional Linear Predictive Coding (LPC) suffers from *slope overload*, which occurs at regions where gray level values change abruptly. Experiments dealing with a new predictive compression method [23], were conducted to remedy this problem. A one-dimensional table was used to store exact pixel values for the locations where the prediction error values are outside some range.

In related work [19, 20], a framework for a multi-phase linear predictive scheme aimed at alleviating the effects of the slope overload problem is described. Difficult-to-predict pixels are iteratively eliminated from the coefficient computation and the LPC coefficients are refined accordingly in two phases. In all of these experiments, the traditional linear predictive model was followed, that is the three points immediately above and to the left were used as predictors for estimating each pixel value.

2.4 Comparison of Compression Methods

No image compression technique is a *panacea* that solves the myriad issues associated with different imaging applications. All compression techniques though effective, have their pros and cons. Prediction based methods tend to be much faster than Transform or Vector Quantization schemes and can be easily realized in hardware [17]. Transform based methods better preserve subjective image quality and are the most insensitive to channel transmission noise. Vector Quantization methods require a complex coder, but offer a simple decoding scheme consisting only of a look-up-table. Pyramidal schemes are suitable for progressive image transmission.

Chapter 3

Description of ASPEN

Existing compression schemes work by either transforming the image (JPEG, wavelets), or by direct statistical redundancy removal (GIF). Each of these schemes has been designed to work in limited domains, such as *'lossy or lossless'* compression of *'binary, grayscale or colour'* images. Unlike its counterparts, *Adaptive Sampling with Predictive Encoding* (ASPEN) is a general purpose image compression scheme which is essentially based on the Predictive Compression model and Detail-Based Segmentation of images. ASPEN's functionality provides for either lossless or lossy compression. It is capable of adapting to different types of images and the particular needs of a given application. Fig 3.1 gives an overview of ASPEN.

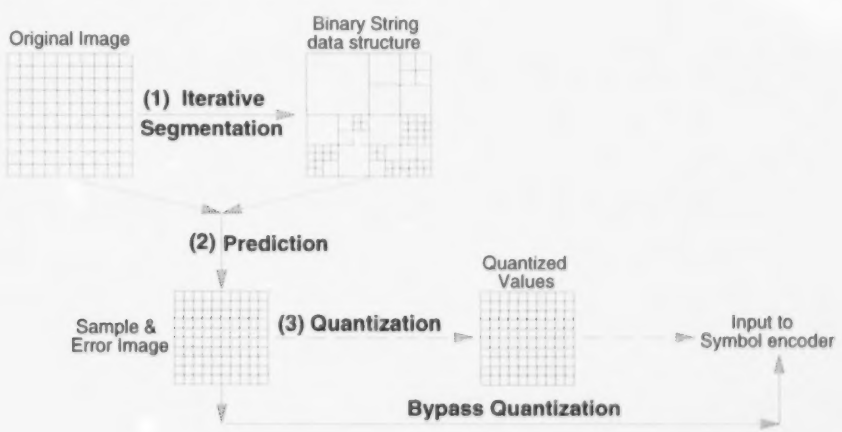


Figure 3.1: **Overview of Adaptive Sampling with Predictive Encoding**

Some salient characteristics of ASPEN are,

- The first stage, *Iterative Segmentation*, attempts to sample areas of the image based on their *detail-content*. Segmentation of the image may be rectangular or square and is represented efficiently by a binary-string based data structure.
- Sample points obtained in this stage are then used to the predict remainder of the image, using prediction schemes ranging from Coefficient based to Bi-linear and Splinal interpolation. Upon prediction, the error values (actual - predicted) along with the sample points are stored in the *sample & error* image.
- Finally, Uniform, Adaptive, or Optimal quantization schemes may be used to roundoff each error value to one of a smaller number of possible values. Output from the quantizer is ready to be coded using efficient symbol encoders such as Huffman or Arithmetic. The quantization scheme may be bypassed, which leads to lossless compression.

In the following sections, *Iterative Segmentation* as well as various prediction and quantization techniques are described in detail.

3.1 Iterative Segmentation

Quite often images are composed of areas with varying *detail-content*. A high *detail-content* implies that the region has rapidly varying gray levels, whereas regions with low *detail-content* have almost constant or slowly varying gray levels. By iteratively segmenting regions of the image based on their *detail-content*, ASPEN attempts to select a set of sample points, which are devoid of the redundancy caused by high correlation present in image data.

For the purposes of segmentation, ASPEN considers grayscale images as surfaces over square or rectangular domains, represented by a set of ordered triples (x, y, z), where (x, y) is the location of a pixel in the image, and $z = I(x, y)$ is the value of the graylevel at that pixel.

The *Iterative Segmentation* algorithm is:

Iterative_Segmentation($I, horz, vert, threshold$)

Input

I– The input image to be segmented.

$horz$– Initial sampling rate in the horizontal (x) direction.

$vert$– Initial sampling rate in the vertical (y) direction.

$threshold$– Threshold value to check segmentation.

Global Variables

$height, width$– Dimensions of the input image I.

Local Variables

$number_of_iterations$– The total number of iterations to be performed.

x, y, k– For local processing.

Iterative_Segmentation()

$number_of_iterations = \log_2(MAX(horz, vert))$.

for($k = 0; k < number_of_iterations; k++$) {

for($y = 0; y < height; y += vert$) {

for($x = 0; x < width; x += horz$)

block = $[(x, y), (x + horz, y), (x, y + vert), (x + horz, y + vert)]$

if ((**block** previously marked $segment_further$) or ($k = 0$)) {

sample the four corner points of the **block**.

test **block** for further segmentation.

mark the **block** appropriately.

}

}

$vert = vert/2$;

$horz = horz/2$;

}

Return

test_block()

approximate predetermined pixels in the block.

for (all approximated pixels)

if $((approximated_value - actual_value) >= threshold)$ then

return $(segment_further)$

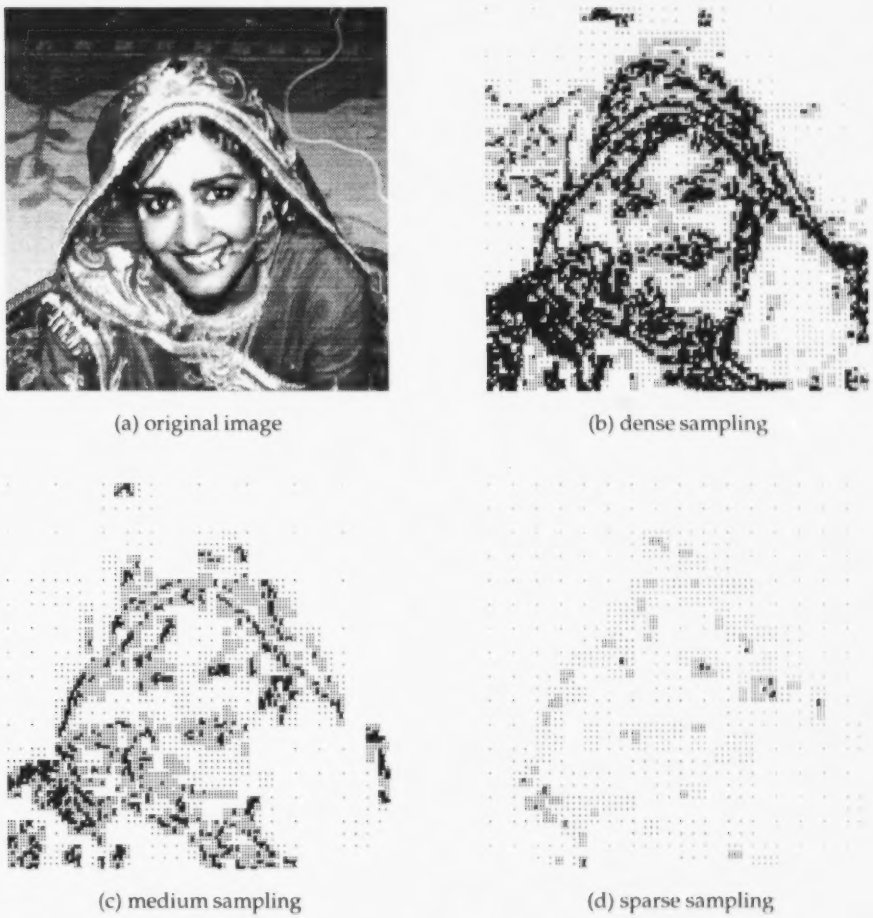
End.

Some salient points of the algorithm are:

- The $number_of_iterations$ in this algorithm is determined by the greater of the two initial sampling rates (horizontal and vertical). For instance, if the greater rate is 32, then there will be $log_2(32)$ or 5 iterations, one each at rate 32, 16, 8, 4 and 2.
- Each iteration segments the image into $((height \cdot width)/(horz \cdot vert))$ number of blocks. Shape of a block depends on the initial sampling rates and may be square or rectangular. A block is represented by its four corner pixels,

  $[(x, y), (x + horz, y), (x, y + vert), (x + horz, y + vert)]$.
- For each block, it is first determined whether it is a sub-block of a block previously marked for segmentation (except in the first iteration). If it is, corner points of the block are then sampled.
- The block is then *tested* for its detail-content. In this phase, previously determined pixels in the block are approximated using the four corner points of the block. The approximation method can be determined by the user at runtime. If the approximation error is more than the *threshold* specified by the user, the block is marked *segment_further*. Otherwise it is not necessary to segment the block further, as the approximation errors lie within the acceptable range.
- The sampling rates are then divided by 2 and the process is repeated.

3.1.1 Demonstration of Sampling



(a) original image

(b) dense sampling

(c) medium sampling

(d) sparse sampling

Figure 3.2: **Demonstration of sampling**

Parameters such as horizontal & vertical rates, threshold can be varied by the user to control the sampling (segmentation) of the image. **Sparse, Medium** to **Dense** sampling can be achieved. An example is presented in Fig 3.2.

(a) - is the input image to be sampled.

(b) - horz = vert = 64, threshold = 20 - *dense sampling*.

(c) - horz = vert = 32, threshold = 40 - *medium sampling*.

(d) - horz = vert = 16, threshold = 80 - *sparse sampling*.

3.1.2 Data Structure to Represent Segmentation

A *binary-string* based data structure has been designed, to represent the segmentation of the image in a compact manner. A schematic of the data structure is presented in Fig 3.3.

Binary String representing this segmentation is:

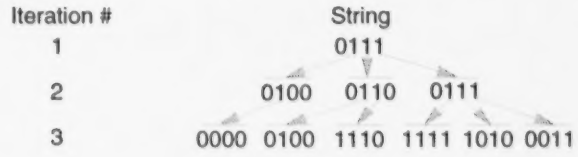


Figure 3.3: **Data Structure to represent Segmentation.**

If a block is to be segmented further, a '1' is appended to the string, otherwise a '0' is appended and the next block is processed. This binary string follows the same order as the segmentation itself, as described in the Iterative_Segmentation algorithm - from left to right first and then top to bottom. In order to reconstruct exact segmentation of the image, all that is required are the original dimensions of the image, initial horizontal and vertical rates and the binary string.

3.2 Prediction Schemes

The *Iterative_Segmentation* scheme, segments the image into square or rectangular blocks, each of which has a low *detail-content*. Each block, which is represented by its four corner pixels, is then approximated using one of the prediction methods, as

shown in Fig 3.4. The prediction method to be used can be determined by the user at runtime.

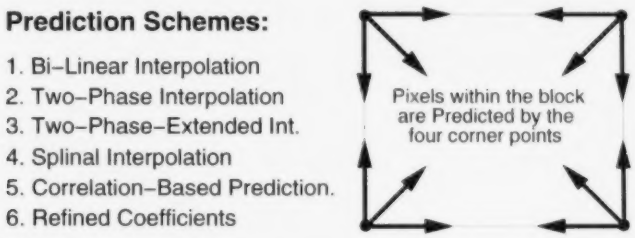


Figure 3.4: **Prediction of a Block.**

In the design of ASPEN, the following prediction schemes were studied.

3.2.1 Bi-Linear Interpolation

If the desired point (which is being predicted) lies within the points being used as predictors, *interpolation* can be used. The simplest interpolation in two dimensions is Bi-Linear Interpolation, which is depicted in Fig 3.5.

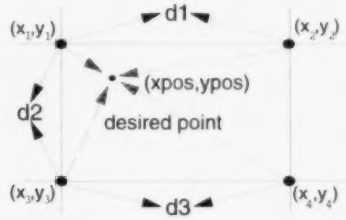


Figure 3.5: **Bi-Linear Interpolation.**

The formula for Bi-Linear Interpolation is,

$$\begin{aligned} \hat{f}(xpos, ypos) &= f(x_1, y_1) + (d1 \cdot xpos) + (d2 \cdot ypos) + (d3 \cdot xpos \cdot ypos). \quad (3.1) \\ d1 &= f(x_2, y_2) - f(x_1, y_1) \\ d2 &= f(x_3, y_3) - f(x_1, y_1) \\ d3 &= f(x_4, y_4) - f(x_3, y_3) \end{aligned}$$

where $\hat{f}(xpos, ypos)$ is the approximated value at the desired point.

The coordinates $(xpos, ypos)$ must lie within the range [0,1]. Their value is calculated relative to the coordinates of the corners of the block.

The *Bi-Linear Interpolation* algorithm is:

Bi-Linear Interpolation$(I, (x_1, y_1), horz, vert)$

Input

$I-$ is the block of pixels to be predicted.

$(x_1, y_1)-$ coordinates of the upper left corner of the block.

$horz-$ width of the block.

$vert-$ height of the block.

Output

$A-$ approximated values for block I.

Local Variables

$(x_2, y_2)-$ coordinates of the lower right corner of the block.

$i, j-$ counter variables.

Bi-Linear Interpolation()

$x_2 = x_1 + horz.$

$y_2 = y_1 + vert.$

for$(i = x_1; i < x_2; i++)$

for$(j = y_1; j < y_2; j++)$

$A(i, j) = bi_linear(I, (i, j), horz, vert)$

Return

The four corner pixels of the block are used to interpolate all points within that block. The surface approximated using Bi-Linear Interpolation is linear along each edge of the block and bilinear in the interior. Bi-Linear Interpolation offers reasonably good results and ease of implementation.

3.2.2 Two-Phase Interpolation

For most approximations, Bi-Linear Interpolation is considered *close enough for government work* [13]. This scheme predicts pixels independent from each other, and in doing

so fails to exploit the correlation present within the pixels in an image. Hence it is not considered very *smart*. A new scheme, Two-Phase Interpolation, has been designed to adapt to the changing scene (variation of gray levels) within a block and thereby take advantage of the interdependency which exists between pixels.

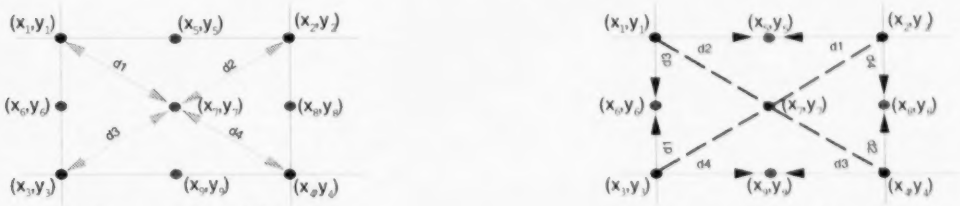


Figure 3.6: **Two-Phase Interpolation.**

The Two-Phase Interpolation algorithm is as follows:

Two-Phase Interpolation$(I, (x_1, y_1), horz, vert)$

Input

$I-$ is the block of pixels to be predicted.

$(x_1, y_1)-$ coordinates of the upper left corner of the block.

$horz, vert-$ dimensions of the block.

Output

$A-$ predicted values for block I.

Local variables

$(x_k, y_k)-$ for local processing, k = 5, 6, 7, 8, 9.

$(d_k)-$ for calculating differences, k = 1, 2, 3, 4.

Two-Phase Interpolation()

if (horz & vert) are smallest possible

then Return.

==== Initializing ====

$x_5 = x_1 + horz/2;\ y_5 = y_1;\ x_6 = x_1;\ y_6 = y_1 + vert/2;$

$x_8 = x_1 + horz;\ y_8 = y_1 + vert/2;\ x_9 = x_1 + horz/2;\ y_9 = y_1 + vert;$

$x_7 = x_1 + horz/2;\ y_7 = y_1 + vert/2;$

==== First Phase – Predict Central Pixel and Calculate Differences ====

$A(x_7, y_7) = bi_linear(I, (x_7, y_7), horz, vert);$

$d_k = abs(f(x_7, y_7) - f(x_k, y_k));\ k = 1, 2, 3, 4.$

==== Second Phase – Predict Rest ====

$\hat{p}(5) = (x_1 \cdot d_2 + x_2 \cdot d_1)/(d_1 + d_2);\ \hat{p}(6) = (x_1 \cdot d_3 + x_3 \cdot d_1)/(d_1 + d_3);$

$\hat{p}(8) = (x_2 \cdot d_4 + x_4 \cdot d_2)/(d_2 + d_4);\ \hat{p}(9) = (x_3 \cdot d_4 + x_4 \cdot d_3)/(d_3 + d_4);$

==== Recursively predict the sub-blocks ====

Two-Phase Interpolation$(I, (x_1, y_1), horz/2, vert/2);$

Two-Phase Interpolation$(I, (x_5, y_5), horz/2, vert/2);$

Two-Phase Interpolation$(I, (x_6, y_6), horz/2, vert/2);$

Two-Phase Interpolation$(I, (x_7, y_7), horz/2, vert/2);$

Return

Some salient steps in the Two-Phase Algorithm are:

- In the first phase, the central pixel (x_7, y_7) of the block, is predicted from the four corner points, using bi_linear interpolation. Differences between the four corner pixel values and the actual central pixel value, are then calculated. These differences give a good indication of the direction in which the pixel values within the block are skewed. This information is then utilized for better prediction of the remaining pixels in the block.
- In the second phase, four other pixels $((x_5, y_5), (x_6, y_6), (x_8, y_8), (x_9, y_9))$ within the block are predicted using the difference values. The block is then subdivided into four equal-sized blocks, each of which is then approximated recursively, using the same procedure.

3.2.3 Two-Phase-Extended Interpolation – (2PE)

Two-Phase-Extended Interpolation, as the name suggests, is an extension of the Two-Phase scheme, differing from the original only in the way it calculates the *difference-*

values.

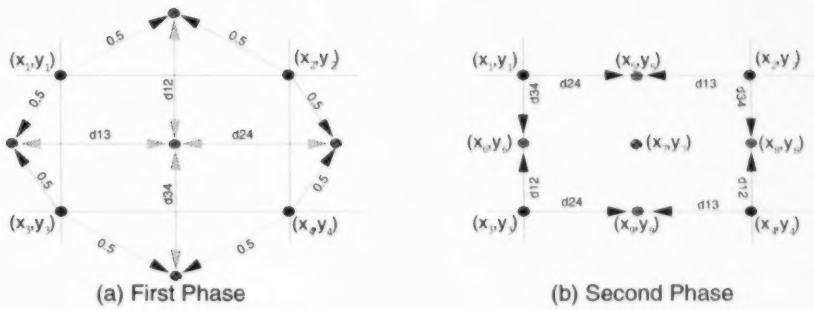


Figure 3.7: **Two-Phase-Extended Interpolation.**

Unlike the Two-Phase scheme, *two* corner pixels are used for calculating each *difference value* in the 2PE model. These *difference values* are then used to predict the remaining pixels in the block, as shown in Fig 3.7. For instance,

$$d_{13} = Abs(f(x_7, y_7) - (0.5 \cdot (f(x_1, y_1) + f(x_3, y_3)))). \quad (3.2)$$

$$d_{24} = Abs(f(x_7, y_7) - (0.5 \cdot (f(x_2, y_2) + f(x_4, y_4)))). \quad (3.3)$$

$$A(x_5, y_5) = (d_{24} \cdot f(x_1, y_1) + d_{13} \cdot f(x_2, y_2))/(d_{13} + d_{24}). \quad (3.4)$$

where d_{13}, d_{24} are *difference values* calculated using corner pixels and the actual value for the central pixel. $A(x_5, y_5)$ is the predicted value calculated using the *difference values* and two corner pixels.

The premise of 2PE is that, taking more corner pixel values into consideration while calculating the *difference values*, should give a better indication of the distribution of pixel values within the block. This information in turn should provide for a better approximation method.

3.2.4 Splinal Interpolation

Splinal Interpolation attempts to fit a smooth curve between the sample points, whereas Linear Interpolation fits a straight line through them [7]. This difference is depicted in Fig 3.8. In some continous-tone images, where gray levels vary gradually, splinal interpolation scheme may be able to better model the image surface.

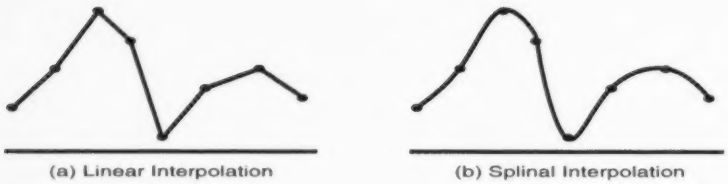


Figure 3.8: **Difference between Linear and splinal Interpolation.**

In this implementation of ASPEN, Quadratic function spline is used to model the image surface. This function is defined by the formula.

$$f(x,y) = C_0 + C_1x + C_2y + C_3x^2 + C_4xy + C_5y^2. \tag{3.5}$$

where C_i 's are the coefficients that uniquely define the quadratic function, and $f(x,y)$ is the value of the function at (x,y). In order to calculate value of the C_i 's, at least six sample points are required. This system of linear equations must be solved,

$$\begin{pmatrix} 1 & x_0 & y_0 & x_0^2 & x_0y_0 & y_0^2 \\ & & \vdots & & & \\ 1 & x_5 & y_5 & x_5^2 & x_5y_5 & y_5^2 \end{pmatrix} \begin{pmatrix} C_0 \\ \vdots \\ C_5 \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_5 \end{pmatrix}$$

ASPEN samples six points from each block, which are then used to calculate the unique coefficient values. Using these coefficients the remaining pixels within the block are approximated.

3.2.5 Correlation-based Prediction

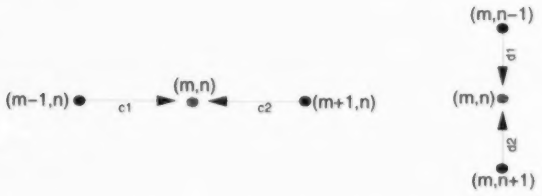


Figure 3.9: **Coefficient based Prediction.**

Correlation-based Prediction is a *smart* prediction scheme, that gathers information from the image and then uses it to predict the image more efficiently. It calculates four

coefficients (C_1, C_2, D_1, D_2), two each for the horizontal and vertical directions, based on the interpixel redundancy present in images. These coefficients are calculated for each possible block size in the image. Individual blocks in the image can then be predicted using these coefficients in the following manner.

$$\hat{f}(m,n) = f(m-1,n)\cdot C_1 + f(m+1,n)\cdot C_2.(horizontal) \tag{3.6}$$

$$\hat{f}(m,n) = f(m,n-1)\cdot D_1 + f(m,n+1)\cdot D_2.(vertical) \tag{3.7}$$

The formulae for calculating these coefficients are,

$$C_1 = \frac{R(m,n,m-1,n)\cdot R(m+1,n,m+1,n) - R(m-1,n,m+1,n)\cdot R(m,n,m+1,n)}{R(m-1,n,m-1,n)\cdot R(m+1,n,m+1,n) - R(m-1,n,m+1,n)\cdot R(m-1,n,m+1,n)}$$

$$C_2 = \frac{R(m-1,n,m-1,n)\cdot R(m,n,m+1,n) - R(m-1,n,m+1,n)\cdot R(m,n,m-1,n)}{R(m-1,n,m-1,n)\cdot R(m+1,n,m+1,n) - R(m-1,n,m+1,n)\cdot R(m-1,n,m+1,n)}$$

$$D_1 = \frac{R(m,n-1,m,n)\cdot R(m,n+1,m,n+1) - R(m,n-1,m,n+1)\cdot R(m,n,m,n+1)}{R(m,n-1,m,n-1)\cdot R(m,n+1,m,n+1) - R(m,n-1,m,n+1)\cdot R(m,n-1,m,n+1)}$$

$$D_2 = \frac{R(m,n-1,m,n-1)\cdot R(m,n,m,n+1) - R(m,n-1,m,n+1)\cdot R(m,n-1,m,n)}{R(m,n-1,m,n-1)\cdot R(m,n+1,m,n+1) - R(m,n-1,m,n+1)\cdot R(m,n-1,m,n+1)}$$

where $R(x_1, y_1, x_2, y_2)$ is the autocorrelation function of the real random to which the image belongs, and is defined as

$$R(x_1,y_1,x_2,y_2) = \sum f(x_1,y_1)\cdot f(x_2,y_2) \tag{3.8}$$

A proof for the formulae of C_1 and C_2 is presented here. Similar proofs can be obtained for D_1 and D_2.

$\hat{f}(m,n) = f(m-1,n)\cdot C_1 + f(m+1,n)\cdot C_2.$

$e^2 = \sum \left[f(m,n) - \hat{f}(m,n)\right]^2$ (error over the entire image)

$e^2 = \sum \left[f(m,n) - \{f(m-1,n)\cdot C_1 + f(m+1,n)\cdot C_2\}\right]^2$

C_1 and C_2 must be calculated such that the error is minimum, hence the first derivative with respect to the coefficients is equated to zero.

$\frac{de^2}{dC_1} = \sum \left[2\cdot\{-f(m,n) + C_1\cdot f(m-1,n) + C_2\cdot f(m+1,n)\}\cdot f(m-1,n)\right] = 0.$

$-\sum \left[f(m,n)\cdot f(m-1,n)\right] + C_1\sum f^2(m-1,n) + C_2\sum \left[f(m+1,n)\cdot f(m-1,n)\right] = 0.$

$\Rightarrow \sum [f(m,n) \cdot f(m-1,n)] = C_1 \cdot \sum f^2(m-1,n) + C_2 \sum [f(m+1,n) \cdot f(m-1,n)]$.

Using equation 3.8, we get

$$R(m,n,m-1,n) = C_1 \cdot R(m-1,n,m-1,n) + C_2 \cdot R(m+1,n,m-1,n). \quad (3.9)$$

Similarly for C_2

$\frac{de^2}{dC_2} = \sum [2 \cdot \{-f(m,n) + C_1 \cdot f(m-1,n) + C_2 \cdot f(m+1,n)\} \cdot f(m+1,n)] = 0.$

$-\sum [f(m,n) \cdot f(m+1,n)] + C_2 \sum f^2(m+1,n) + C_1 \sum [f(m-1,n) \cdot f(m+1,n)] = 0.$

$\Rightarrow \sum [f(m,n) \cdot f(m+1,n)] = C_2 \cdot \sum f^2(m+1,n) + C_1 \sum [f(m-1,n) \cdot f(m+1,n)]$.

Using equation 3.8, we get

$$R(m,n,m+1,n) = C_1 \cdot R(m-1,n,m+1,n) + C_2 \cdot R(m+1,n,m+1,n) \quad (3.10)$$

Solving the two equations 3.9 and 3.10, with two unknowns C_1 and C_2, we get the previously defined values for C_1 and C_2. Hence the proof.

3.2.6 Refined Coefficients

In the correlation-based prediction scheme, coefficients for each block size are calculated using the entire image. However, due to the nature of the segmentation in ASPEN, not all pixels in the image play a role in the prediction of each block size, as shown in Fig 3.10 .

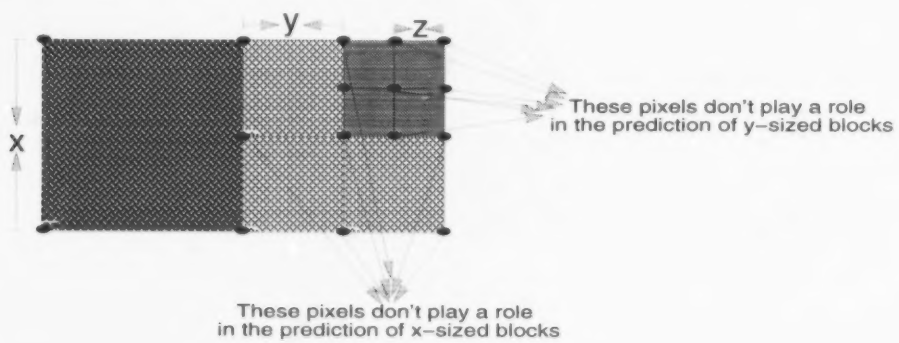


Figure 3.10: **Refined Coefficient.**

In the Refined Coefficients scheme, only pixels that play a role in the prediction of a particular block size are used in the calculation of coefficients. For example, the pixels

that define smaller sized blocks don't play a role in the prediction of bigger blocks, and hence should not be used in the calculation of coefficients for these blocks. Intuitively the refined scheme should exhibit an improvement over the previous coefficient calculation method.

3.3 Quantization

A quantizer is a device whose output has only a limited number of possible values. Each input is forced to take one of the allowable output values. Quantization is an irreversible process because, given the output value, it is impossible to determine the exact input value.

Uniform (equal) and Optimal (unequal or tapered) quantization are the two methods that exist for quantizing data values. Both these schemes have been implemented in ASPEN. Another scheme, Adaptive quantization, that takes advantage of the ASPEN-segmented image has also been developed. The user may specify the preferred quantization scheme at run time.

3.3.1 Uniform Quantization

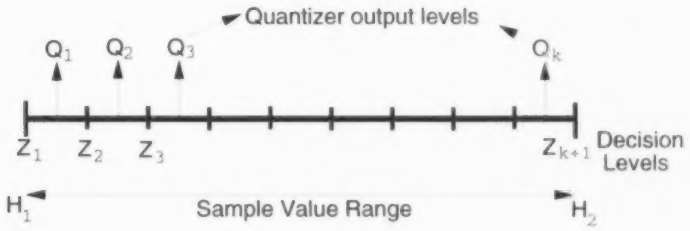


Figure 3.11: **Uniform Quantization.**

Image pixel values are said to be uniformly distributed if they occur with equal frequency in the image. Uniform quantization, which divides the input range into equal sized blocks, is the best and the least complex solution for such a situation. As is shown in Fig 3.11, the input range H_1, H_2 is divided into K equal sized blocks $(Z_1, Z_2, ..., Z_{k+1})$. The quantizer output levels $(Q_1, Q_2, ..., Q_k)$ lie midway between the decision levels as

is shown in the following formulae.

$$Z_k = (Q_{k-1} + Q_k)/2. \tag{3.11}$$

$$Q_k = (Z_k + Z_{k+1})/2. \tag{3.12}$$

3.3.2 Optimal Quantization

When the sample values in a certain range occur frequently while others occur rarely, it would be more efficient to use quantization levels that are finely spaced inside this range and coarsely spaced outside it. This would increase the average accuracy of quantization without increasing the total number of levels.

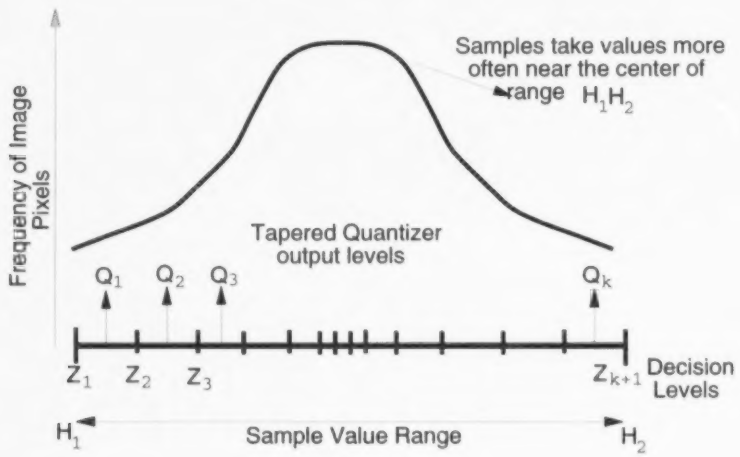


Figure 3.12: **Optimal Quantization.**

For a given number K of output levels, the decision levels Z_k and output levels Q_k must be determined such that the mean square quantization error δ_q^2 is a minimum.

$$\delta_q^2 = \sum_{k=1}^{K} \int_{Z_k}^{Z_{k+1}} (z - q_k)^2 \cdot p(z) dz \tag{3.13}$$

where $p(z)$ is the probability density function for the input sample. To minimize δ_q^2 for a given K, partial derivatives of δ_q^2 with respect to Z_k and Q_k are equated to zero. The derived formulae for Z_k and Q_k are:

$$Z_k = (Q_{k-1} + Q_k)/2. \tag{3.14}$$

$$Q_k = \left[\int_{Z_k}^{Z_{k+1}} z \cdot p(z)dz\right] / \left[\int_{Z_k}^{Z_{k+1}} p(z)dz\right]. \quad (3.15)$$

For an optimum quantizer, the decision levels (Z_k) are located halfway between the output levels (Q_k) while each Z_k is the centroid of portion of $p(z)$ between (Z_k) and (Z_{k+1}).

Steps involved in optimally quantizing a given input sample are:

- Pick an appropriate value for Q_1.
- Then calculate the succeeding Z_k's and Q_k's by using equations 3.14 and 3.15.
- If the calculated value of the last output level Q_k is equal to the centroid of the area between Z_k and Z_{k+1}, then it is the correct solution.
- If Q_k is not equal to the appropriate centroid, the calculation must be repeated with a different value of Q_1.

The search for the correct value of Q_1 can be systematized so as to yield correct results in a short time. A number of different ways to design and implement optimal quantizers can be found in [10, 24].

3.3.3 Optimal versus Uniform Quantization

Table 3.1 presents results from the comparison of Uniform with Optimal Quantization. Tests were conducted for image *Lena* at quantization levels 2, 4, 8 and 16.

Number of Levels	Uniform	Optimal	Percentage
2	3.94	4.40	11.67
4	7.45	9.77	31.14
8	14.62	17.40	19.01
16	28.42	29.99	5.52

Table 3.1: **Optimal versus Uniform Quantization**

Number of Levels is the number of levels the image is quantized into. *Uniform* and *Optimal* represent the SNR (Signal-to-Noise Ratio) values for Uniform and Optimal quantizations, respectively. *Percentage* represents the amount by which Optimal outperforms Uniform. Examples from these tests are presented in Fig 3.13.

(a) orig. image (b) equal - 2 (c) optimal - 2

(d) equal - 4 (e) optimal - 4 (f) equal - 8

(g) optimal - 8 (h) equal - 16 (ı) optimal - 16

Figure 3.13: **Examples of Optimal and Uniform Quantization**

3.3.4 Adaptive Quantization

Both Uniform and Optimal Quantization schemes treat the image to be quantized as a *single-entity*, and therefore are not able to take advantage of the segmentation produced by ASPEN. However, in some cases it may be more efficient to treat blocks of the image differently. A new scheme, Adaptive Quantization has been designed to accomplish it in an efficient manner.

ASPEN segments the image into square or rectangular blocks of varying sizes, each of which is at least locally smooth and thus may be predicted within reasonable error.

Error values within these blocks vary considerably, and are primarily dependent on the size of the block. Since the corner pixels used for prediction in a large block lie further away, it is found that bigger blocks have larger errors compared to blocks of smaller size.

Figure 3.14: **Adaptive Quantization.**

To take advantage of this fact, Adaptive Quantization deals with each segmented block separately, quantizing it based on its size. Bigger blocks are quantized into fewer levels as the error values contained in them are larger, whereas smaller blocks need to be quantized densely.

Chapter 4

Search in Parameter Space

A key feature of ASPEN is that it can be customized to meet the specific needs of a user. In order to achieve this facility, a user can directly input the parameters that control the functioning of ASPEN, thereby acquiring the desired compression and image quality. ASPEN parameters can take a wide range of values; a detailed description follows,

- **horz-rate & vert-rate**

  These are the initial sampling rates in the horizontal and vertical directions, respectively. These rates must be powers of 2, and are selected from the range $\in \{4, 8, 16, 32, 64, 128\}$.

- **upper-thresh & lower-thresh**

  These represent the threshold values used to check segmentation at the first and the last iterations, respectively. The values of these thresholds are picked from the range $\in \{0, 255\}$.

- **num-points-to-check**

  This parameter decides the number of points that will be approximated when a block is being tested for further segmentation. Values it can take are $\{5, 9\}$.

- **prediction-method**

  Prediction scheme to be used is determined by this parameter. One of the following schemes may be used, { bi-linear, two-phase, two-phase-ext, corr-coeffs,

refined-coeffs, spinal }.

- **quantization-method**

  This parameter determines the quantization to be used. Choices are { optimal, adaptive, equal }.

- **num-levels**

  This determines the number of output levels of the quantizer. Set of values lies in the range $\{0, 256\}$.

Experienced users of ASPEN, might want to experiment with the parameter values in order to achieve higher performance. ASPEN allows them this freedom. However, a novice user (lacking the experience to vary parameter values) might be satisfied with a *good* setting for the parameters, at which ASPEN performs at or near its peak. In order to choose a *good* setting for the parameters, a systematic *search of the ASPEN parameter space* was conducted. Fifteen images representative of several different image types were used in the experiments. The search results presenting the *good* settings of ASPEN parameters, are reported as a guideline for future users. However, the user can easily alter any of these parameters to achieve higher performance in a particular application.

4.1 First Phase – Wide Range Search

The parameters *quantization-method* and *num-levels*, relate to lossy compression. They do not have any effect on the *sampling of the image* or the *quality of prediction*, and hence, it was decided to leave them out of the experiments for parameter search.

In the first phase, ASPEN was run over a *wide range* (covering most of the allowable range) of parameter values. The results obtained were quite comprehensive and demonstrated a distinct trend in ASPEN's performance. They helped locate a *useful range of values* that the parameters should take. The values that the input parameters took in these runs are presented in Table 4.1,

Parameters	Values
Horizontal & Vertical rates	8, 16, 32, 64, 128.
Upper & lower thresholds	0, 50, 100, 150.
Num_points_to_check	5, 9.
Prediction_method	All six schemes.

Table 4.1: **Setting of ASPEN parameters in the First Phase**

There are 5x5x4x4x2x6 = 4800 possible combinations of these parameter values, and ASPEN was run once for each setting. To cover the wide range of image types (facial, satellite, aerial, landscapes, close-ups, generated), 15 representative images were picked for testing. The results obtained were tabulated; each table contained 400 numbers (25x16 - for the rate and threshold combinations). Twelve such tables were obtained for each image (2x6: 2 for the num-points-to-check and 6 for each of the prediction schemes).

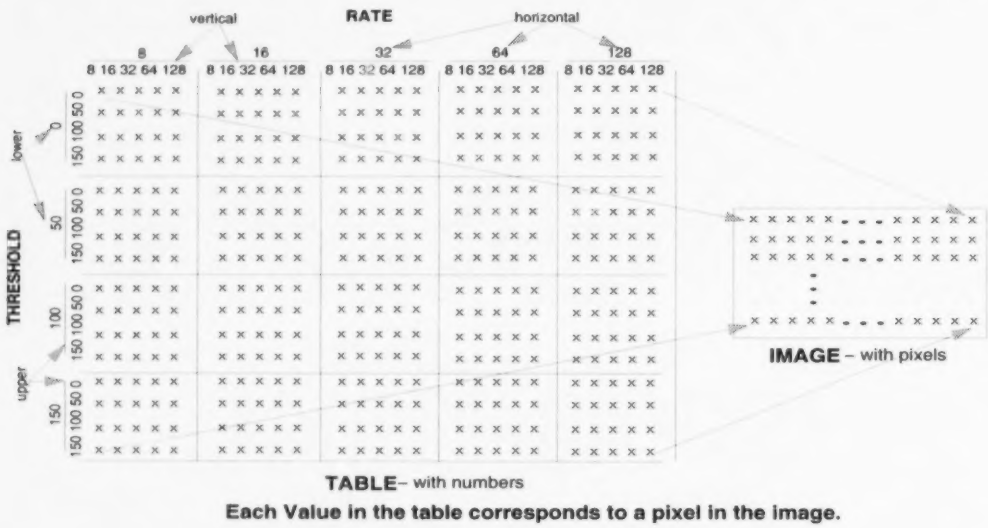


Figure 4.1: **Table represented as an image**

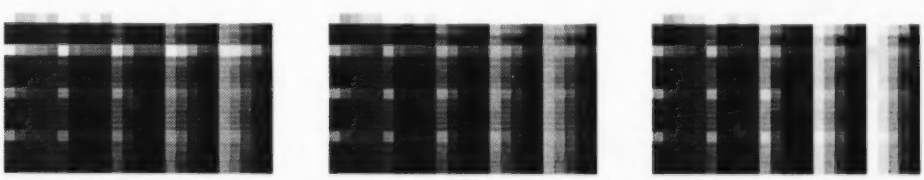
For demonstration purposes, it was decided to present the information contained in the tables using grayscale images. Visual information (an image with light intensities) is more effective in demonstrating the trend exhibited by ASPEN's parameters (shown in Fig 4.1).

The transformation of information from the tables to images was achieved in the following way.

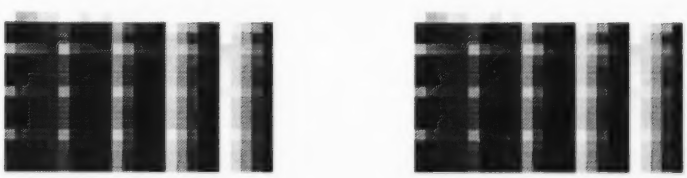
- The maximum and the minimum values in the tables were first detected. They were then assigned grey level values at the extremes of the 8-bit grey level scale, that is 0 and 255.
- Remaining values in the table were then assigned a grey level value, corresponding to their distance from the maximum and the minimum values, as presented in Formula 4.1. In this way a light intensity value is obtained for each number in the table.

$$Graylevel(\text{for table value X}) = \frac{X - min}{max - min} \cdot 256. \tag{4.1}$$

Results for three images, obtained in the first phase, are presented in Figures 4.2, 4.3, 4.4. Search results were similar for all images. Each figure contains five pseudo images, one each for the five prediction schemes - Two-Phase, Two-Phase-Ext, Bi-Linear, Corr. Based and Refined Coeffs. All images represent 9-point division. Darker areas in these pseudo images represent better performance of ASPEN.

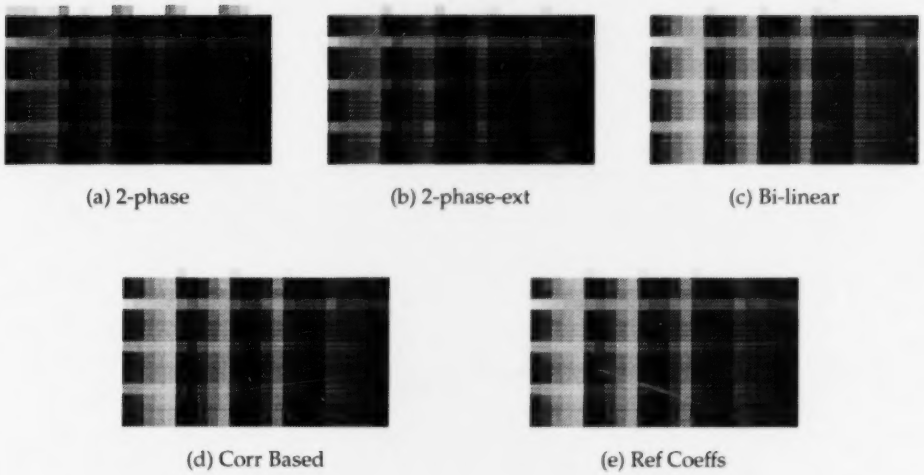


(a) 2-phase (b) 2-phase-ext (c) Bi-linear



(d) Corr Based (e) Ref Coeffs

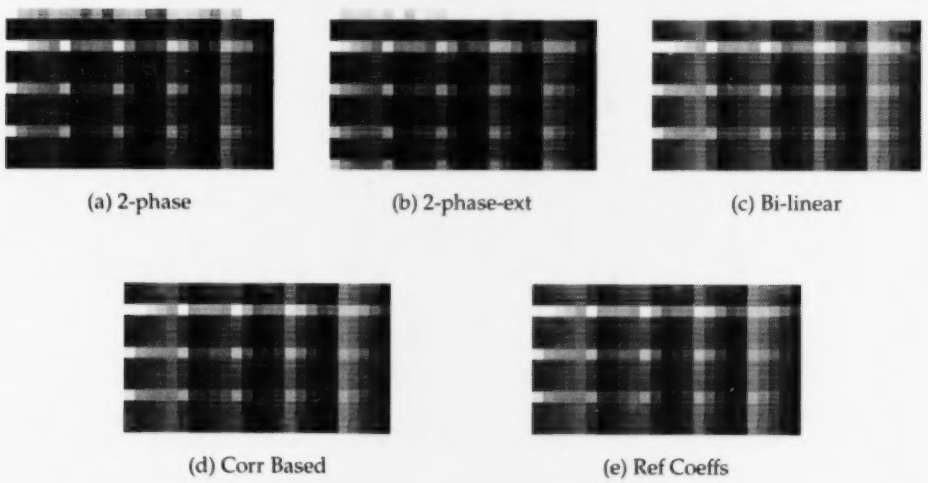
Figure 4.2: **Demonstration of Trend – Image1 – Wide Search**



(a) 2-phase (b) 2-phase-ext (c) Bi-linear

(d) Corr Based (e) Ref Coeffs

Figure 4.3: **Demonstration of Trend – Image2 – Wide Search**



(a) 2-phase (b) 2-phase-ext (c) Bi-linear

(d) Corr Based (e) Ref Coeffs

Figure 4.4: **Demonstration of Trend – Image3 – Wide Search**

Results indicated that ASPEN performs better when the sampling rates are (32, 64, 128). At low sampling rates more points are sampled, which in turn reduces the level of ASPEN's performance. It was also noticed that performance is better if the horizontal and vertical sampling rates are equal (square segmentation). The optimum range for the thresholds was narrowed down to 50-150. Lower threshold values represent lesser

prediction and more sampling. Performance of ASPEN at 5 and 9-point division is quite close, however, for most cases 9-point division gave slightly better results. Of all the **Prediction Schemes** tested, it was observed that Two-Phase and Bi-Linear gave the best results, followed closely by the Two-Phase-extended scheme.

4.2 Second Phase – Fine Tuning

The second set of runs were designed to *fine-tune* the parameters, that is focus into the exact setting where ASPEN's performance is *good* (at or near its peak). The values that the input parameters took in these runs are presented in Table 4.2.

Parameters	Values
Horizontal & Vertical rates	32, 64, 128.
Upper & lower thresholds	50, 75, 100, 125, 150.
Num_points_to_check	9.
Prediction_method	Two-Phase, Bi-Linear, Two-Phase-Extended.

Table 4.2: **Setting of ASPEN parameters in the Second Phase**

ASPEN was run 375 times for each image (once for each parameter setting = 5x5x3x3). The results obtained were tabulated; each table contained 125 numbers. Three such tables were obtained for each image (one each for the three different prediction schemes located in the first phase).

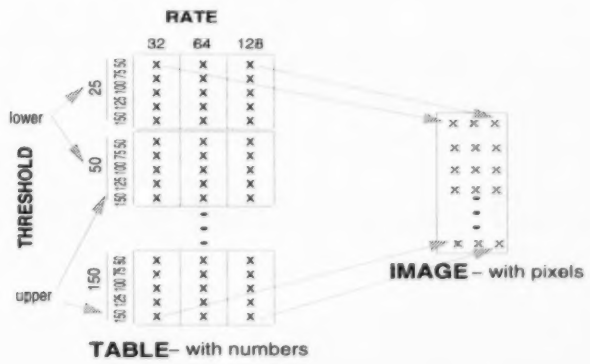


Figure 4.5: **Table represented as an image**

Again for demonstration purposes, the information contained in tables is presented

in visual form (pseudo images), as is shown in Fig 4.5. Results for the three images, obtained in the second phase, is presented in Fig 4.6 (a), (b), (c) represent the results for Image - 1, (d), (e), (f) for Image -2, and (g), (h), (i) for Image - 3.

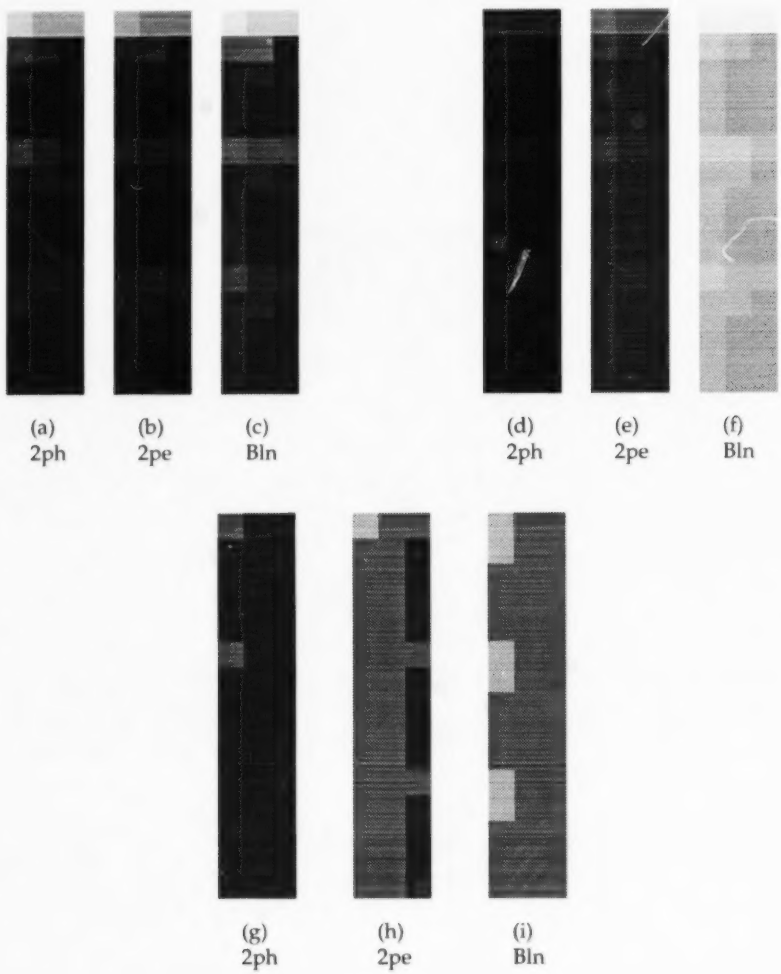


Figure 4.6: **Demonstration of Trend – Fine Tuning**

Although the performance at rates 32, 64 and 128 is quite close, it was observed that rates 32 and 64 were consistently better. *Good* range for the thresholds is still 50-150, however, it was detected that ASPEN performs better, whenever the value of lower threshold is greater than the value of higher threshold. The user can use any value

between 50 and 150 for the thresholds. Also, Two-Phase scheme emerged as the best prediction method. The optimal setting for ASPEN parameters is depicted in Tab 4.3.

Parameters	**Values**
Horizontal & Vertical rates	32, 64.
Upper & lower thresholds	50-150 (lower > Upper).
Prediction_method	Two-Phase.

Table 4.3: **Optimal Setting of ASPEN parameters**

Chapter 5

Performance Comparisons

Since ASPEN can be used for both Lossless and Lossy compression, its performance was tested and compared with the most widely used standards in these areas, Linear Predictive Coding (lossless) and JPEG (lossy). Several measures can be used to evaluate compression algorithms, and the merits of some are described first. Finally, results from the performance comparisons are presented.

5.1 Performance Measures

The performance of compression algorithms is highly image dependant. Several criteria can be used to measure the performance of compression algorithms, and the relative importance of each depends on the needs of the application (for which the compression algorithm is required).

Most commonly used measures are *compression ratio, image quality* and *speed of compression/decompression*. These measures are easily quantifiable. Some other criteria that cannot be directly measured are *consistency of compression, ease of hardware implementation* and *standardization*.

5.1.1 Compression Ratio

Compression ratio is the most significant criterion used for measuring the performance of compression algorithms. It is simply the percentage of savings in the number of

bits required to represent the data. For most algorithms it is difficult to determine the compression ratio beforehand, as it depends on the individual characteristics of the image being compressed. Images with greater detail generally give lower compression ratios. Compression ratio K is,

$$K = b/H. \tag{5.1}$$

where b is the least number of bits needed to represent the image quantization levels and H is the entropy of the image. Entropy gives a lower bound on the average number of bits required to code a set of input values, which is closely achievable by Huffman [3]. It is evident from formula 5.1, that if entropy of an image is known, compression ratio can be calculated easily.

Since this version of ASPEN was intended to be a prototype, a symbol coder such as Huffman or Arithmetic was not implemented. Therefore, the actual storage required for ASPEN compressed images was not calculated. Instead entropy of the error image was calculated which gives an upper bound on the amount of compression possible (closely achievable by Huffman). Hence entropy has been used to substitute compression ratio in all the results presented.

5.1.2 Fidelity Criteria

Fidelity criteria provide a means of quantifying the nature and extent of information loss due to compression. These criteria are applicable only for lossy compression schemes.

If the loss of information can be expressed as a function of the original and the decompressed image, it is said to be based on an *objective fidelity criterion*. Examples of such criteria are *mean-absolute-error, root-mean-square-error* and *signal-to-noise-ratio.*

Let $f(x, y)$ represent an input image and $\hat{f}(x, y)$ denote the decompressed image. Let M and N be the dimensions of the image. For any value of x and y, the error $e(x, y)$ is defined as,

$$e(x, y) = \hat{f}(x, y) - f(x, y). \tag{5.2}$$

This forms the basis of all *objective fidelity criteria.*

The *root-mean-square-error* is the square root of the squared error averaged over the M x N array, or

$$e_{rms} = \left[\frac{1}{M \cdot N} \cdot \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} e(x,y)^2 \right]^{1/2} . \tag{5.3}$$

A characteristic of the mean squared error is that it accentuates large errors in individual pixels. A closely related measure, the mean absolute error may be used instead,

$$e_{abs} = \frac{1}{M \cdot N} \cdot \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} |e(x,y)| . \tag{5.4}$$

The *signal-to-noise-ratio* is the most widely used measure for image quality. For the decompressed image it is defined as,

$$SNR_{rms} = \left[\frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \hat{f}(x,y)^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [e(x,y)]^2} \right]^{1/2} . \tag{5.5}$$

The numerator is the output or the decompressed signal. The denominator term is the error in a pixel within the image and has a value of zero for the pixels that are not altered during compression.

Objective fidelity criteria offer a simple and convenient mechanism for evaluating information loss. However, evaluating image quality through a human observer may be more appropriate, as most decompressed images are ultimately viewed by human beings. Such evaluations are said to be based on *subjective fidelity criteria* and may be accomplished by showing the decompressed image to a cross section of viewers and averaging their evaluations [17]. However, individual opinion of the quality of a particular image varies considerably, and accurate results based on subjective testing are difficult to obtain. In this thesis, only *objective fidelity criteria* are considered.

5.1.3 Compression Time

Many imaging applications such as teleconferencing, require real-time processing and transmission of images. Compression/Decompression time is the key factor that determines whether any algorithm is suitable for applications with such stringent requirements.

A number of variables affect the time required to compress/decompress images using any algorithm. They may be the *particular implementation, platform being used* or *time required for I/O*. Implementation of algorithms on hardware generally outperform software implementations.

This version of ASPEN was intended to be a prototype. It is not a fully functional image compression/decompression system yet. Many memory and speed optimizations will have to be performed in the future versions. For that reason, this thesis does not present any results based on compression time. It must be noted that Prediction based methods (like ASPEN) tend to be much faster than Transform based schemes and can be easily realized in hardware [17].

5.2 Characteristics of the Tests

ASPEN was tested and compared with Linear Predictive Coding (LPC) and JPEG. Some salient characteristics of the tests were,

- **Environment**

  A Sun SPARC station SLC under Unix was the environment used for the tests and comparisons.

- **Image File Format**

  Raster file format was chosen as the image format for the storage of both uncompressed and compressed ASPEN images.

- **Implementation**

  ASPEN, LPC and JPEG were implemented in the C programming language. All programs are platform independent.

- **JPEG**

  Data for JPEG compression was obtained using the free software of the Independent JPEG group. The IJG software provides control over the compression size and quality of the image, through the use of a parameter Q - *for quality*. Q must

be in the range 0-100, with 100 corresponding to the highest image quality and the lowest compression ratio.

- **Test Images**

  Fifteen grayscale images (size 256 x 256) were selected for the comparison tests. The images are listed according to their types in Table 5.1. Sample images from each type are presented in Fig 5.1.

Image Type	Names of the Images
Facial	Lena, Pfeifer, Monalisa, Model
Aerial	Building
Close Ups	Flower, Car, Fruit
Generated	Window
Landscapes	Carrier, Bay, Street, Sunset, Potala
Satellite	Earth

Table 5.1: **Different Images used for Testing**

(a) Facial (b) Aerial (c) Close Ups

(d) Generated (e) Landscape (f) Satellite

Figure 5.1: **Examples of Images**

- **ASPEN Parameters**

  In all the tests performed, value of the input parameters in ASPEN remained

constant. These values were determined in the extensive search of ASPEN parameter space (shown in Table 5.2). *Two-phase* was the prediction scheme used

H. Rate	V. Rate	Lower Th.	Upper Th.	Division
32	32	50	150	9 point

Table 5.2: **Setting of ASPEN parameters for comparisons**

in the lossless comparisons of ASPEN with LPC, whereas the *bi-linear* scheme was used in the lossy comparisons with JPEG. *Uniform* quantization scheme was used in the lossy comparisons.

5.3 ASPEN vs LPC

Table 5.3 presents results from the comparison of ASPEN with LPC in terms of entropy.

Image Name	Orig. Ent.	L P C	A S P E N	Fared	Percentage
lena	7.44	5.18	5.22	-	0.53
pfeifer	7.75	5.70	5.52	+	2.32
monalisa	6.62	5.30	5.20	+	1.51
model	6.06	4.36	4.20	+	2.64
flower	7.19	4.51	4.64	-	1.80
fruit	7.14	4.77	4.80	-	0.42
car	7.15	5.23	5.17	+	0.84
carrier	5.43	5.73	5.39	+	6.26
sunset	5.85	4.73	4.40	+	5.64
window	3.41	3.51	2.50	+	29.61
potala	5.15	4.77	4.36	+	7.96
bay	7.15	5.54	5.49	+	0.69
building	6.36	5.85	5.95	-	1.72
earth	5.95	4.60	4.51	+	1.51
street	7.25	5.27	5.29	-	0.27

Table 5.3: **Comparison of ASPEN and LPC**

The column headings are interpreted as:

- **Image Name:** Name of the image.

- **Orig. Ent.:** Entropy of the original image. **LPC:** Entropy of the LPC compressed image. **ASPEN:** Entropy of the ASPEN compressed image.
- **Fared:** A + sign indicates that ASPEN fared better, whereas a - sign shows that LPC was the better scheme.
- **Percentage:** A number indicating the percentage by which either ASPEN or LPC was better. The formula used was,

$$percentage = \frac{|ASPEN - LPC|}{Orig.Ent.} \cdot 100. \tag{5.6}$$

5.3.1 Analysis of the Results: ASPEN vs LPC

Of the 15 images tested, ASPEN fared better than LPC in 10 of them. For this set of images, average compression achieved by LPC was 1.272, whereas average ASPEN compression was 1.325.

Based on this set of images, it appears that ASPEN outperforms LPC. The images for which ASPEN performed better, it beat LPC by a considerable margin. On the other hand LPC beat ASPEN only marginally in some of the images. It should also be noted that the parameter settings of ASPEN used in these tests may not be optimal for each individual image. A higher compression ratio can be obtained for each image after experimenting with the parameters (customizing the parameters for each image).

5.4 ASPEN vs JPEG

Table 5.4 presents the results from the comparison of ASPEN with JPEG (with bi-linear prediction and uniform quantization). The column headings are interpreted as,

- **Image Name:** Name of the image.
- **Orig. Ent.:** Entropy of the original image.
- **JPEG:** SNR and Entropy of the JPEG compressed image.
- **ASPEN:** SNR and Entropy of the ASPEN compressed image.

- **Quality:** The Quality percentage (Q) at which ASPEN beat JPEG.

Image Name	Orig. Ent.	JPEG		ASPEN		Quality
		snr	*ent*	*snr*	*ent*	
lena	7.44	25.61	1.19	28.57	1.20	70
pfeifer	7.75	21.58	0.93	24.30	0.86	50
monalisa	6.62	16.49	0.74	18.76	0.74	50
model	6.06	35.24	0.85	35.27	0.70	80
flower	7.19	36.39	1.20	36.12	1.16	80
fruit	7.14	21.31	1.01	24.94	0.97	70
car	7.15	48.17	1.89	48.32	1.77	88
carrier	5.43	16.79	0.63	17.31	0.62	50
sunset	5.85	29.28	0.93	30.32	0.91	80
potala	5.15	40.72	1.10	42.68	1.02	88
bay	7.15	20.53	1.21	22.79	1.22	70
building	6.36	18.29	1.72	19.30	1.74	75
street	7.25	23.81	1.15	25.99	1.16	70
window*	3.41	121.07	1.30	34.75	1.30	95
earth*	5.95	68.59	2.60	58.32	2.58	95

Table 5.4: **Comparison of ASPEN and JPEG**

5.4.1 Analysis of the Results: ASPEN vs JPEG

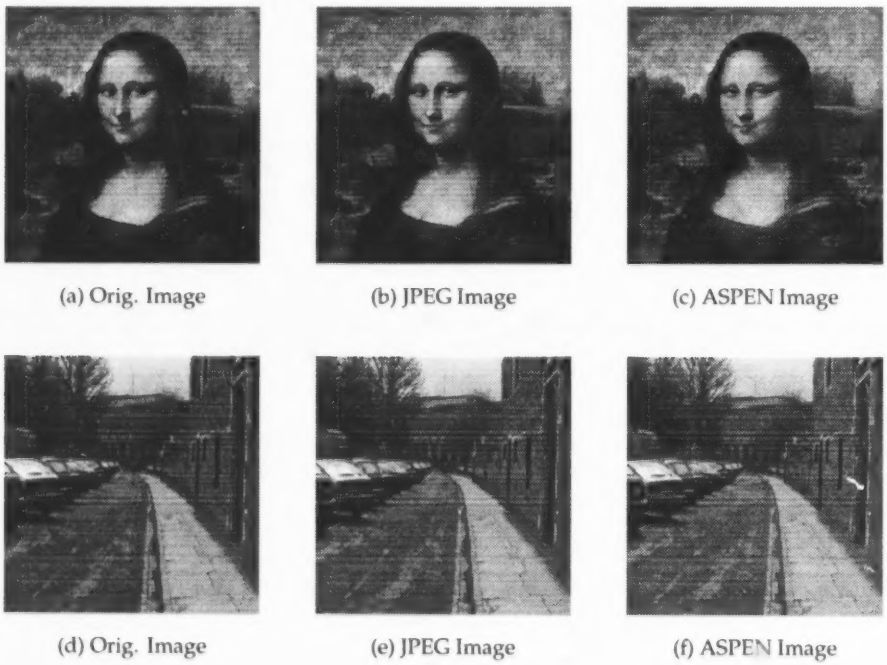
Of the fifteen images tested, ASPEN did not beat 2 images. It beat JPEG at Q-88 in 86%, at Q-80 in 73%, at Q-75 in 53%, at Q-70 in 46% and at Q-50 in 20% of the images. Average value of SNR in JPEG was 27.24, whereas average SNR value of ASPEN was

28.82. Average entropy achieved by JPEG was 1.12, whereas average entropy achieved by ASPEN was 1.08.

Based on this set of images, it appears that ASPEN scheme using Bi-linear prediction and Uniform quantization outperforms JPEG if the goal is to achieve low compression (that is ratios between 2:1 and 10:1) while maintaining image quality as measured by SNR. As the compression ratio increases, however, JPEG compression is clearly better of the two methods. Hence if image quality is essential, ASPEN scheme is better, while if small file size is the goal, JPEG is superior.

5.5 Examples

Some examples of images compressed using ASPEN and JPEG are shown in Fig 5.2. The *monalisa* image was compressed at Quality level 50 and the *street* image was compressed at Q 80 in JPEG.



(a) Orig. Image (b) JPEG Image (c) ASPEN Image

(d) Orig. Image (e) JPEG Image (f) ASPEN Image

Figure 5.2: **Examples of Compression – ASPEN and JPEG**

Chapter 6

Possible Applications of ASPEN

ASPEN is a *general purpose* image compression scheme that can be used to compress different images within several applications. For images requiring lossless compression such as medical images to teleconferencing images for lossy compression, ASPEN is capable of adapting to each image and particular needs of the application.

However, the scope of ASPEN extends beyond being a *general purpose* compression scheme. Some imaging applications for which it is particularly suited, are described in this chapter.

6.1 Scanning an Image Database

When an operator is searching an image database looking for a particular image, (in the absence of indexing information), it is desirable to quickly display a low quality version of each candidate image. It is not necessary to have the highest possible image quality to find the image for which the operator is looking as lower resolution may be sufficient to reject/accept an image. Fig 6.1 shows an image compressed using JPEG and ASPEN at high compression rates,

- **(a)** - Original grayscale image.
- **(b)** - JPEG image, compression ratio is 49.54.
- **(c)** - ASPEN image, compression ratio is 51.75.

(a) Original Image

(b) JPEG Image

(c) ASPEN Image

Figure 6.1: **Blockiness in JPEG at low compression levels**

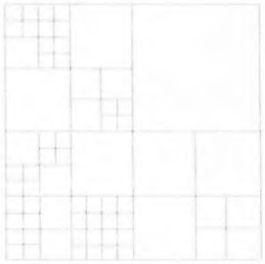
The JPEG image is quite *blocky*, which makes it hard to recognize the original image. At a comparable compression ratio, the ASPEN image has a much better visual quality and is therefore more suited for such an application. It must be noted that the quality of the JPEG image as measured by SNR is higher than the ASPEN image.

6.2 Combining Image Encryption with Compression

In an image database, security is a major issue which is just as important as storage. This is especially true in a wireless environment, where transmission of data demands a high ratio of compression and a high degree of security at the same time.

Existing Image Encryption techniques require extra storage, which affects the compression ratio significantly. Furthermore, these schemes treat image encryption separately from compression, which requires additional time and space in the process.

To recover the segmentation of the image, ASPEN has a sophisticated reconstruction mechanism. The *Key* for reconstruction is treelike data structure which can be stored as a binary string (as shown in Fig 6.2).



Binary String representing this segmentation is:
1011 1001 1011 0001 1101 0001 0110 1010 1111 0000

Figure 6.2: **Key for Image Reconstruction**

This key is at most $1/10^{th}$ of the image in size. Various coding schemes can be used to record or transmit this key. The images compressed or transmitted in this way would be inherently secure in the sense that the only way to reconstruct the images is to obtain and understand the key.

Hence, ASPEN can be used as an Image Encryption scheme, in which the security is an intrinsic component of the compressed image.

6.3 Redundancy Sieving

In ASPEN, parameters such as the sampling rates and threshold can be varied by the user to control the amount of *detail-retention*, or in other words, *redundancy-removal* from the original image. The removal of redundancy from the image is referred to as *Redundancy Sieving*. Areas of low detail content (of constant or slowly varying gray levels) are filtered out and only pixels representing areas of high detail content remain in the redundancy-sieved image. An example is presented in Fig 6.3.

(a) Original Image (b) Redundancy Sieved Image

Figure 6.3: **Redundancy Sieving in ASPEN**

An image reconstructed using only the sample points from the *redundancy-sieved* image (all error values are made zero), would be of high quality in the areas of detail. Possible applications that might need such images, are Weather Forecasting, Tracking, Arial Combat, Teleconferencing, and Topographical Analysis. In such applications, where one needs to focus on areas of detail, a system that can automatically sieve out redundancy in the images, will be most beneficial. ASPEN provides such a functionality.

6.4 Like-Image Database

In an image database, most images are of the same type. For example, a hospital lab may store hundreds of chest x-ray images, which bear great similarity to each other. A police station may store thousands of face photos, each of which is a front-view of a human head. In any of these cases, an image compressor will routinely deal with a great number of similar images. Existing algorithms compress images from *scratch* and fail to utilize knowledge about particular image types.

ASPEN's parameters (using knowledge or after experimentation) can be optimized for a particular image type to obtain the maximum possible compression. Therefore, ASPEN is suited for compression of images in a database where most or all of the images would be of a particular type.

Chapter 7

Conclusions

This thesis has presented a *general purpose* image compression scheme – *Adaptive Sampling with Predictive Encoding* (ASPEN), which can be customized to individual images or the particular needs of an application. Its functionality also extends to providing both Lossless or Lossy compression. These qualities make ASPEN an attractive image compression system.

ASPEN is based on the Predictive Compression model. Six prediction schemes were implemented and tested thoroughly. Results showed that *Two-phase* prediction works best for lossless compression, whereas if lossy compression is desired, *Bi-linear* performs the best. Predicted values of pixels in *Two-phase* are interdependent and for this reason, errors contained within one pixel propagate onto others. This problem does not arise in the lossless case where pixels are not quantized and hence contain no errors.

Based on the image set used in our experiments, it appears that ASPEN outperforms LPC for lossless compression.

For lossy compression, if the goal is to achieve low compression (that is ratios between 2:1 and 10:1 – or upto 70-75% Q in JPEG) while maintaining *good* image quality, then ASPEN should be used instead of JPEG. As the compression ratio increases, however, JPEG compression is clearly better of the two methods. Hence if image quality is essential, ASPEN scheme is better, while if small file size is the goal, JPEG is superior.

Some applications for which ASPEN is particularly suited are *Scanning an Image Database, Image Encryption* and *Redundancy Sieving*. Study showed that ASPEN's functionality provides well for all the requirements of these applications.

7.1 Suggestions for Future Work

The first course of action, is to take ASPEN from its prototype stage, to a complete compression/decompression system. To accomplish this, an efficient symbol coder such as Huffman or Arithmetic will have to be implemented. Also many speed and memory optimizations will have to be performed for the final version of ASPEN.

Contrary to traditional Image Encryption schemes, ASPEN treats compression and encryption together, thereby reducing time and space requirements. The use of a binary string based structure in ASPEN (for image reconstruction), makes security an intrinsic component of the compressed image. The application of ASPEN for Image Encryption shows promise and needs to be investigated further.

So far ASPEN has been designed for compression of still-images. The functionality of ASPEN can be easily extended to motion picture compression. Successive frames may be used to transmit previously unsampled points, thereby constantly improving image quality in all areas of the image. It will be interesting to see how ASPEN fares against the existing motion picture compression standards such as MPEG.

A quadratic spline based prediction scheme was experimented with. Results showed that this particular function was not able to model the image surface effectively. Some other functions such as exponential, might be able to better model the images, and their performance should be investigated.

The study of ASPEN's performance with respect to Optimal Quantization is in experimental stages, at the time of this writing. Preliminary results show that Optimal quantization outperforms Uniform quantization by about 15-20%. Once fully functional, optimal quantization will give ASPEN the edge it needs to beat JPEG at lower quality (Q) levels.

One can envision a knowledge-based ASPEN system in the future, that can system-

atically generate and utilize *knowledge* from an image. It can also accumulate *knowledge* while processing and compressing images. Such a system would have the ability to understand images and use that knowledge to automatically select the best setting of ASPEN parameters. This ability will make the functionality of ASPEN completely transparent to its users.

Bibliography

[1] D. G. Daut and D. Zhao. Improved dpcm algorithm for image data compression. *Image Processing Algorithms and Techniques.*, pages 199–210, 1990.

[2] M. Gerla. High speed local area networks. *IEEE Spectrum*, 28(8):26–31, August 1991.

[3] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley, 1992.

[4] R. M. Gray. Vector quantization. *IEEE Acoustics, Speech and Signal Processing magazine.*, 1(2):4–29, 1984.

[5] D. Hanson. Progress in fiber optic LAN and MAN standards. *IEEE LCS Magazine*, pages 17–25, May 1990.

[6] D. A. Huffman. A method for the construction of minimum redundancy codes. *Proc. IRE*, 40(10):1098–1101, 1952.

[7] E. Isaacson and H. B. Keller. *Analysis of Numerical Methods*. New York – -Wiley, 1966.

[8] A. Jacquin. Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Transactions on Image Processing*, 1(1):18–30, January 1992.

[9] M. Kunt, A. Ikonomopolous, and M. Kocher. Second generation image-coding techniques. *Proceedings of IEEE.*, 73(4):549–574, April 1985.

[10] J. Max. Quantizing for minimum distortion. *IRE Transactions on Information Theory*, pages 7–12, 1960.

[11] A. N. Netravali. *Digital Pictures: Representation and Compression.* Plenum Press, New York, 1988.

[12] W. P. Pennebaker and J. L. Mitchell. *JPEG – Still Image Data Compression Standard.* Van Nostrand Reinhold, 1993.

[13] W. H. Press. *Numerical Recipies in C.* Cambridge University Press, 1992.

[14] S. A. Rajala, M. R. Sivanlar, and W. M. Lee. A second generation image coding technique using human visual system based segmentation. *Proceedings of IEEE Int. Conf. on Acoustics, Speech and Signal Processing.*, 73(5):436–440, May 1982.

[15] R. P. Rao and W. A. Pearlman. On entropy of pyramid structures. *IEEE Trans. on Information Theory.*, 37(2):407–413, 1991.

[16] A. Rosenfeld and A. Kak. *Digital Picture Processing.* Academic Press, 1976.

[17] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis and Machine Vision.* Chapman and Hall, 1993.

[18] J. Turner. New directions in communications. *IEEE Journal on Selected Areas in Communications*, 8(8):1439–1447, October 1990.

[19] F. Uludamar and X. Li. An improved linear predictive image compression method. *Proc. of CIPS Edmonton*, pages 23–32, 1988.

[20] F. Uludamar and X. Li. Some experiments in multi-phase iterative predictive compression of pictures. *Proc. of International Conference on Computing and Information*, pages 515–519, 1989.

[21] G. K. Wallace. The jpeg still image compression standard. *Communications of the ACM.*, April 1991.

[22] T. Welch. A technique for high performance data compression. *IEEE Computer*, 17(6):645–669, June 1984.

[23] J. Wetherill and X. Li. Experiments on an improved predictive image compression method. *Proc. of CIPS Edmonton*, pages 361–369, 1987.

[24] R. C. Wood. On optimum quantization. *IRE Transactions on Information Theory*, pages 248–252, 1969.

[25] H. Zailu and W. Taxiao. Mdpcm picture coding. *IEEE Int. Symp. on Circuits and Systems.*, pages 3252–3255, 1990.